2R−6

概念モデル上のカーディナリティに着目した仕様の洗練化

横田和久 橋本正明 岡本克巳 佐藤正和 竹中豊文
ATR 通信システム研究所

1 はじめに

筆者らは概念モデルとしてERモデルと制約を適用した仕様記述言語PSDL(Program Specification Description Language)[1]を提案し、PSDLで記述された仕様をCプログラムに変換するためのPSDLコンパイラを研究中である[2]。さらに、PSDLで記述された仕様を再利用して、以下に示すステップでソフトウェアを自動作成する方法を研究中である[3]。

1. たとえば、販売在庫管理業務のような対象世界をPSDLで記述して、それを再利用対象のモデルシステム仕様として蓄積する。
2. モデルシステム仕様をユーザーの要求に合わせてカストマイズし、目的システム仕様を得る。
3. その目的システム仕様を洗練化し効率的なプログラム仕様を抽出する。
4. そのプログラム仕様をPSDLコンパイラでプログラムへ変換する。

上記の第3ステップにおいて、目的システム仕様を洗練化するためにはプログラムの入出力データを指定して、その入出力データ間のデータフローパスを選択することが必要である。

パス選択問題について従来[4]等の研究があるが、本稿ではERモデルのカーディナリティに着目したパス選択法を提案する。

2 PSDLとモデルシステム仕様

ERモデルでは対象世界の「もの」や「こと」をエンティティと呼び、そのエンティティの間の対応をリレーションシップと呼ぶ。個々のエンティティの性質はアトリビュートの値で表される。

PSDLでは、ERモデルで定められるエンティティタイプとアトリビュート、リレーションシップタイプを記述し、さらにそれらの間に成り立つ以下の3種の制約を記述する。

1. アトリビュート値を得るためのアトリビュート値従属性制約
2. エンティティを得るためのエンティティ存在従属性制約
3. リレーションシップを得るためのリレーションシップ存在従属性制約

ところで、リレーションシップタイプにはカーディナリティが記述されており、以下に示すようにエンティティとリレーションシップとの間の数量関係が表されている。

(1) 1つのエンティティに1つのリレーションシップがつながる。

(-1) 1つのエンティティに多くても1つのリレーションシップしかつながらない。

(M) 1つのエンティティに0からM(>1)までのリレーションシップがつながる。

なお、入出力データとエンティティタイプの間にも上記と同じ数量関係を規定するためのカーディナリティを記述する。

以上に述べたPSDLで、図1に示すモデルシステムの詳細レベルの仕様が表される。なお、モデルシステム仕様を理解しやすくし、仕様の選択肢も表すため、図1のマクロレベルの仕様も定める。

Specification Refinement by Cardinalities in Conceptual Model.
Kazuhisa Yokota, Masaaki Hashimoto, Katsumi Okamoto, Kazumasa Sato, Toyofumi Takenaka
ATR Communication Systems Research Laboratories

3 データフローパス選択による仕様の洗練化

モデルシステム仕様から仕様の選択、修正および追加を行いカストマイズして得られた目的システム仕様からデータフローを示す有向グラフを作成する。図2に示すように有向グラフの節点はエンティティタイプやアトリビュート、リレーションシップタイプ、制約に対応し、有向枝は制約によって生じるデータフローを表す。

その有向グラフの上で、指定されたプログラム入出力データ間で必要となるデータフローパスを選択して、そのパス上の仕様を抽出することにより仕様の洗練化を行う。

以下、有向枝の合流節点に着目したパスの選択法を示す。

3.1 制約節点の有向枝選択

制約を表す式(または関数)の中にアトリビュート値の参照が2つ以上あると、制約節点へ有向枝が合流する。この場合、式(または関数)の計算にどのアトリビュート値も欠かすことが出来ないので、全ての有向枝を選択する。

3.2 エンティティタイプ節点の有向枝選択

エンティティが2つ以上のエンティティ存在従属性制約、および入力データによりエンティティタイプ節点に有向枝が合流する。以下合流する有効枝が2本の場合について、選択法を示す。なお、3本以上への拡張も同様である。2本の有向枝のカーディナリティを(m, n)で表現する。

1. **カーディナリティが(1, 1)**
   合流有向枝のカーディナリティがそれぞれ(1, 1)のときは、同じエンティティの集合がが両方の枝から得られる。このため、エンティティを得るのに片方の有向枝しか必要としないので、指定された入力データにつながっているパス上の有向枝を選択する。
2. **カーディナリティが(1, -1)**
   合流有向枝のカーディナリティがそれぞれ(1, -1)のときは、カーディナリティが -1 の枝から得られるエンティティは全て、カーディナリティが1の枝からも得られるものの部分集合となっている。このため、カーディナリティが 1 の有向枝を選択する。
3. **カーディナリティが(-1, -1)**
   合流有向枝のカーディナリティがそれぞれ -1, -1 のときは、それぞれの枝から異なるエンティティの部分集合が得られる。このため、両方の有向枝を選択する。
4. **カーディナリティがMの場合の補正**
   合流有向枝のカーディナリティがMのときは、流れ込むエンティティから見ると、カーディナリティを -1 とみなすことができる。

3.3 アトリビュート節点の有向枝選択

アトリビュートの値が2つ以上のアトリビュート値従属性制約、および入力データによりアトリビュート節点へ有向枝が合流する。この場合、以下3.2と同様に合流する有効枝が2本の場合について選択法を示す。

1. **カーディナリティが(1, 1)**
   合流有向枝のカーディナリティがそれぞれ(1, 1)のときは、同じアトリビュートの値がそれぞれの枝から得られる。このため、アトリビュート値を得るのに片方の有向枝しか必要としないので、入力データにつながっているパス上の有向枝を選択する。

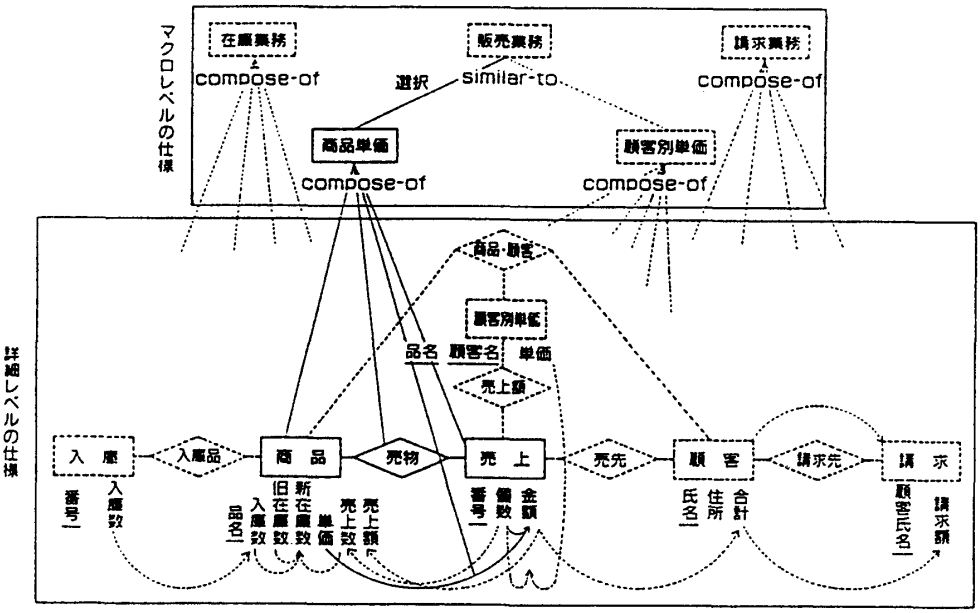

図 1: モデルシステム仕様

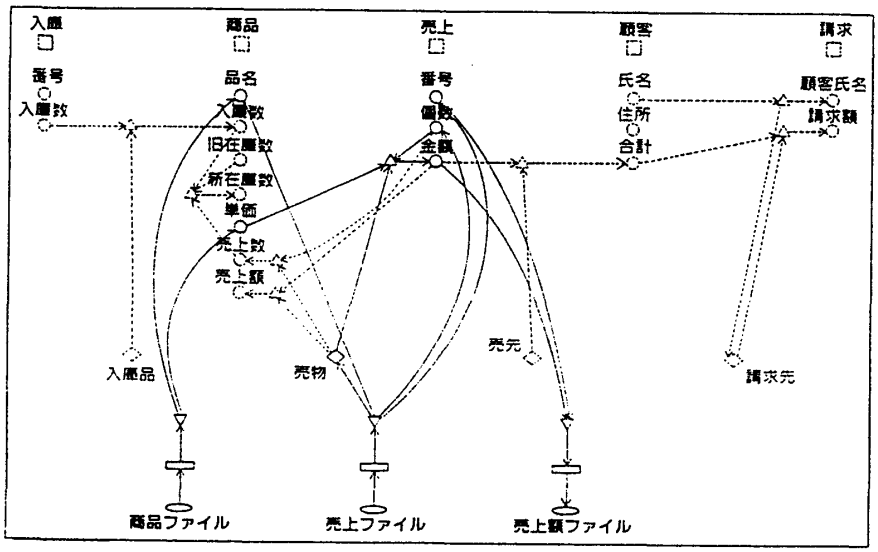

図 2: 有向グラフ

2. **カーディナリティが (1, -1)**
合流有向枝のカーディナリティがそれぞれ (1, -1) のときは、カーディナリティが -1 の枝から得られるアトリビュートの値は全て、カーディナリティが 1 の枝からも得られる。このため、カーディナリティが 1 の有向枝を選択する。

3. **カーディナリティが (-1, -1)**
合流有向枝のカーディナリティがそれぞれ (-1, -1) のときは、それぞれの枝から異なるエンティティのアトリビュート値が得られる。このため、両方の有向枝を選択する。

4. **アグリゲーション関数によるカーディナリティの補正**
合流有向枝の始点となっている制約節点が、例えばサンメンション（総和）のようなアグリゲーション（集合）関数の場合、その枝のカーディナリティが -1 か M であっても、必ずアトリビュート値が得られる。そのため、その枝のカーディナリティを 1 とみなして上記の処理を行なう。

3.4 リレーションシップタイプ節点の有向枝選択

リレーションシップがリレーションシップ従属性情報制約と、入力データの両方から得られる場合に、有向枝が合流する。この場合、必要な枝は自動的には判断できないのでユーザーの選択に任せる。

4 おわりに

本稿に述べたデータフローパス選択法を、ソフトウェア自動作成システムの機能として実現中である。さらに、現在作成中のモデルシステム仕様の事例に対して、本選択法の有効性を実験する予定である。

参考文献

[1] K. Okamoto and M. Hashimoto: On real-time software specification description with a conceptual data model-based language, In *Proc. of ICC'90*, 1990.

[2] M. Hashimoto and K. Okamoto: A set and mapping-based detection and solution method for structure clash between program input and output data, In *Proc. of COMPSAC'90*, 1990.

[3] 岡本克巳, 橋本正明: 制約指向の概念モデルを用いた高次部品化によるプログラム合成, 電子情報通信学会技報ソフトウェアサイエンス 89-18,1989.

[4] Stanley Y. W. Su, Shirish Puranik, and Herman Lan: Heuristic Algorithms for Path Determination in a Semantic Network, In *Proc. of COMPSAC'90*, 1990.